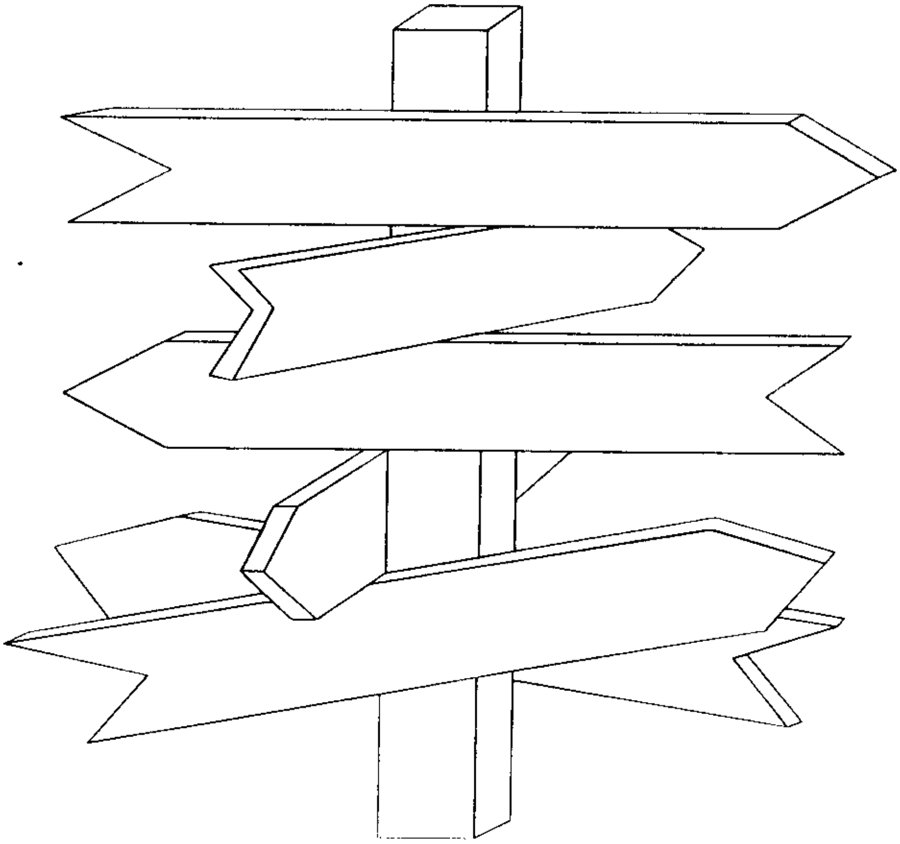
PCVR

Vol 1 Issue 5

Virtual Reality and the IBM Personal Computer



Head Tracking Technology Survey

Head Mounted Display Optics

Build a Boom Head Tracker

Contents

2 **Preface**
Life vs. VR Experiences and survey results.

3 **Survey of Available Head Tracking Technology**
This article describes the different head tracking technologies available.

12 **Building a Boom Head Tracker**
We build a boom head tracker using readily available parts for under $10.00.

21 **Virtual Imagination**
A new column by Steve and Shawn Casey.

23 **Building a Head Mounted Display**
The third article in this series in which, we look at the optics for a head mounted display.

27 **Graphics**
This issue's column discusses rotation about an axis parallel to a coordinate axis.

30 **Working with Rend386**
A new column dedicated to Rend386

35 **The Disk**
What's on the enclosed disk.

36 **Last Page**
Next Issue.

Issue #5
September/October 1992

Issue Prices:
Trial Issues One and Two:
$3.00 US./Canada
$5.00 Overseas

Back Issues:
$4.50 US./Canada
$6.50 Overseas

Subscription Rates:
$26.00 US./Canada
$38.00 Overseas

Advertisement Rates:

Call for current prices.

For more information:
Address:
PCVR
1706 Sherman Hill Rd #A
Laramie, Wyoming 82070

Phone/FAX:
(307) 742 - 7675

Email:
gradecki@rodeo.uwyo.edu

Check, Money Order, Visa/MC accepted.

PCVR is a bi-monthly publication of Gradecki Publishing. Copyright © Joseph D. Gradecki 1992. All rights reserved. Gradecki Publishing accepts no liability for any consequences of its use, and no responsibility for any errors that may appear

Preface

On August 5, 1992, my wife and I had a baby boy. It was the most exhilarating moment I have ever experienced. It was amazing how that little human came into the world. It made me think about Virtual Reality and the degree to which we will be able to simulate reality.

At this moment in time, we can not come anywhere close to simulating our true reality. We can not accurately reproduce our vision, hearing, touch, smell, or taste. We certainly can not simulate that "sixth sense" we seem to experience from time to time.

There is simply no conceivable way to create a Virtual Reality experience that mimics childbirth or any similar life occurrence that we as humans can only appreciate by actually going through it in real life.

SURVEY RESULTS
As of this printing, we have received 12% of the surveys sent out. We appreciate the time you spent filling out and returning the survey. We have come up with several changes to the magazine in both format and content.

1) The format of the magazine will change starting with the seventh issue (Jan/Feb 1993). The format will be more user-friendly and targets us toward a glossy format.

2) The disk will replace all software listings in the magazine from the fourth issue on. We always try to include executables when possible.

3) Many people asked for software. The format for next year will include motion figures (objects with programs attached), a Virtual Reality operating system for REND386 called PCVROS. One important feature of the operating system will be call-in functions. Both modem and network support will be featured.

4) The graphics column will continue with its tutorial.

5) A Rend386 tutorial column will be added starting with this issue.

6) We will create a Virtual Reality Bulletin Board in the near future.

IN CLOSING
We would like to welcome Steve and Shawn Casey to the PCVR team. These gentlemen will be writing a column in each issue of PCVR dedicated to topics of their own desire called Virtual Imagination. Don't Miss It.

Survey of Available Head Tracking Technology

For a Virtual Reality experience to provide a realistic representation of our true world, it must allow the user to move and look around. Each of these actions can be considered a form of head tracking, although moving and looking are different activities. In this article, we explore the many different forms of available head tracking. Some of these forms can be duplicated by the homebrewist and some cannot.

INTRODUCTION

It can be argued that the original intention for head tracking came from the military to simplify the process of flying a jet fighter. The military wanted the pilot to be able to control certain parts of the plane with his head. A pilot could look at a specific part of the cockpit and a computer would activate a switch. The only problem was determining where the pilot was looking. A head tracker would solve this problem.

One of the first companies to create a head tracker was Polhemus Navigation Systems. Since that time, several companies have come up with different ways to track head movement. There are four basic categories for head trackers: mechanical, optical, ultrasonic, and magnetic.

Mechanical

The mechanical head trackers use means other than electronic to track the movement of a head. Three mechanical systems are boom, gryo, and a 'fishing line' system.

BOOM A boom head tracker consists of a series of rods connected together with joints. Each of the joints has a potentiometer to measure the degree of flex. Figure 1 shows a possible setup.

As the user moves about the Virtual World, the potentiometers send analog signals to an Analog-to-Digital converter. The digital information is fed to a computer to calculate the position of the user. The position of the potentiometers determine the degree of freedom the computer can detect.

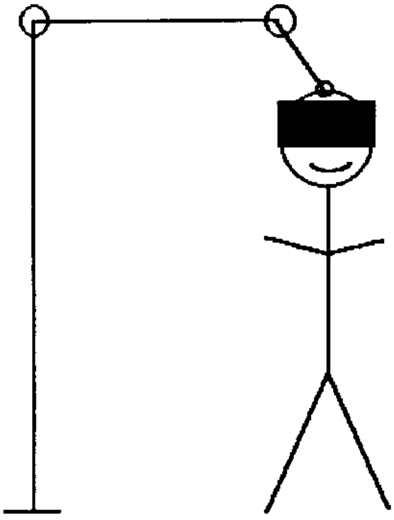


Figure 1

The potentiometer at the base of the boom measures the Z axis freedom. The pots at the first and second joints determine the X axis freedom. There is no direct way to determine the Y axis freedom. These measurements are of the user himself and not their gaze. To determine which direction the user is looking, we add several additional components.

A potentiometer at the end of the boom and attached to the head mounted display will measure the horizontal pan for the user. Using a pot at this location will give the computer a larger degree of accuracy for the pan components of head movement. In my opinion, this is the most important element for the user in a Virtual World.

However, we do not want to overlook the tilt and roll components. To measure these components, we can use two tilt switches. The switches give the computer three measures each for tilt and roll.

The boom head tracker restricts the user since it attaches directly to the user. However, if built correctly the user should have minimum discomfort. The boom allows for instantaneous measurement of the position of the user in three space as well as measurement of their gaze. In a later article we discuss the construction of a boom head tracker.

GYRO Over a year ago, there was a discussion on the Usenet news group, sci.virtual-worlds, about using small gyros to determine the position of the head. These gyros are used in R/C airplanes to keep themselves stable. These gyros measure the amount of rotation from an object. Thus, if we mounted one on our head, we could measure the amount of tilt, roll, and pan. We could not measure the position of our head, where we are in space. Another problem with these gyros is 'continued rotation'. If you panned your head from left to right fast and then tried to panned from right to left, the gyros would still have significant rotation remaining from the left to right pan. The measured results would not be accurate. These gyros would not be an acceptable head tracker because of the 'continued rotation' and the lack of position information.

'STRING TEST' The string test is a restrictive system for measuring either head position or head movement. A system of three strings are attached to the users head. One string is connected to the back, one to the immediate left, and another to the immediate right. Figure 2 illustrates this system.

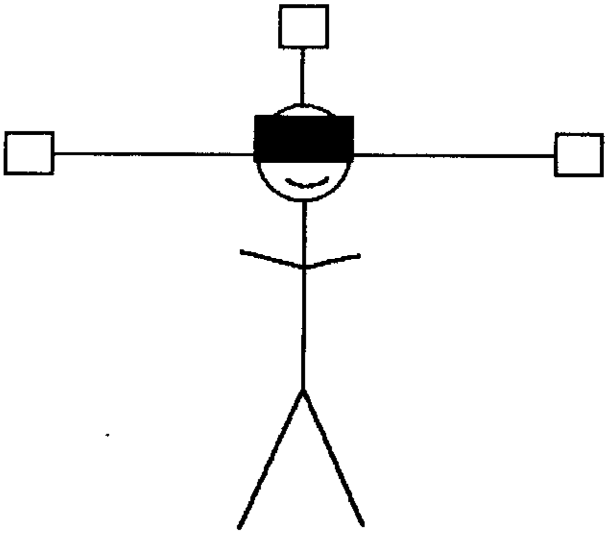


Figure 2

The opposite ends of the strings would be connected to an auto-retrieving reel. This reel would always be trying to retract the string. As the user moves their head, each of the different reels would be either retracting or releasing string. By measuring the initial state of the system and subsequent movements, a computer could determine the position of the user without delay. There are problems with this setup. The first is the user is attached. The second is the string test is limited to either determining the position of the head in three space or the gaze direction of the user. The reason for this can be illustrated by an example using figure 2. If the user walks forward a small amount all three of the reels release string. If the user is back in their original position and turns their head in either direction, all three reels release string. There is no way to differentiate between the two actions.

The system works well for position detection. Left and right movements cause different reels to release strings. Forward and reverse movements cause the back reel to either release or retract string. One additional problem is measuring the amount of release or retraction the reel is experiencing.

Optical

Optical head trackers use a source of light to measure the degree of head movement. This system has been grouped as inward camera and outward camera.

INWARD CAMERA A possible obvious way to determine the position and gaze of a user is to look at them. A computer can look at an object using a camera and a digitizer. A camera, video or CCD, is positioned toward the user. As the user moves, the computer digitizes the pictures from camera and locates differences between the current picture and the previous picture. The computer can determine the new position and gaze of the user by analysis. The differences in the types of inward camera setup is the object the computer analyses. There are two different objects; a visible source on the object or edge detection.

Visible Source The visible source options involves putting some sort of pattern on the head of the user that the computer can easily and quickly recognize and analyze. This form of head tracking is very closely related to computer vision and image processing. Several different schemes have been proposed.

The first involves using a series of infrared LEDs put on the user's head as a triangle. Figure 3 illustrates the placement of the LEDs.

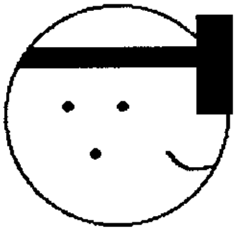


Figure 3

When the user pans his head, the horizontal position of the three LEDs change. When the user tilts his head, the LEDs roll forward or backward. When the user rolls his head, the vertical position of the LEDs change. All of these changes in position can be accurately detected by the computer once an image is digitized.

In addition to the gaze direction, this type of head tracker can measure changes in head position. As the user moves in three space, the visible distance of the LEDs will change between digitized pictures.

It may be obvious by now that the computer is doing a large amount of computations. The steps involved include:

- digitizing the pictures
- locating the LEDs
- comparing their location to the previous picture
- performing calculations to determine X,Y,Z position
- performing calculations to determine gaze direction
- discarding the old picture, saving current

If the computer is doing all of these operations as well as rendering the scene, we will need a personal computer that is still in development. There are two ways to solve this problem. The first is to dedicate a second personal computer to do all of the steps listed above and send the final data to the rendering system. There second solution is to design a hardware card to do all of the steps. Dave Stampe, the author of Rend386, has done this. In a recent posting to the Usenet news group sci.virtual-worlds, he documents his head tracker using the LED method.

One problem with this LED system is the capture area of the camera. Mr. Stampe reports a movement area of 50cm in any direction. For most current Virtual Experiences, this is adequate.

There are two other methods that can be used instead of the LEDs. Research has been performed using a series of patterns on the head. The patterns basically take the place of the LEDs. The principle behind the patterns is easier detection by the computer software. A second approach proposed putting lasers on the head and capturing the room on camera. The lasers would give three points on the walls that the computer would use to determine position. It was noted that the resolution of the camera and digitizing software is probably not good enough to pick up the fineness of the laser points.

Edge Detection Edge detection is very much like the LED approach above. Instead of having LEDs or patterns on the head, the digitizing software picks out the edges of the user's face. This system was developed by three researchers at NTT Human

Laboratories in Japan. The system uses a fast algorithm that picks out the contrast between the user's hair and face to recognize the face. The hardware consists of a video digitizer connected to a SUN4 workstation. The system operates at 10 frames per second. Their application sent the face orientation to an IRIS workstation where a graphics face mimicked the movement of the human user. The system software is written in C and provides 6 DOF under limited conditions.

One problem with the system is the person must have an acceptable amount of hair in order for the system to detect the hair-face lines.

INWARD CAMERA HOMEBREW Building an inward camera head tracker is easier than it may appear. You need the following equipment:

- Personal computer
- Digitizing card
- Camera
- Software

It is assumed that you already have the personal computer. Digitizing cards are available from many different sources. Check the bibliography section for two cards that can be built yourself. The camera can be expensive. If you have a camcorder, you already have the necessary camera. The software for detecting the LEDs can be found in many computer vision and image processing books.

OUTWARD CAMERA The opposite of the inward camera is the outward. This type of head tracker puts some number of cameras on the head of the user. The most visible system of this type was developed at the University of North Carolina - Chapel Hill.

Three infrared sensors are placed on the head of the user in a specific pattern. These sensors have lenses on them to provide for a better visual picture. On the ceiling is a pattern of infrared LEDs. The LEDs are placed in suspended ceiling panels. There is a total of 32 LEDs per panel. The UNC system had a working area of 10' by 12' for a total of 960 LEDs. Figure 4 shows a possible setup.

This system is computationally expensive. The UNC setup used a 68030-based computer to manage the LEDs, another 68030-based computer as a queue manager, and an i860-based computer for collinearity. These three systems feed information to the UNC pixel-plane 5 graphics engine. It was noted in the articles that due to a bus incompatibility, all of the computer systems were needed. If the incompatibility issue was solved, several of the computers could be eliminated.

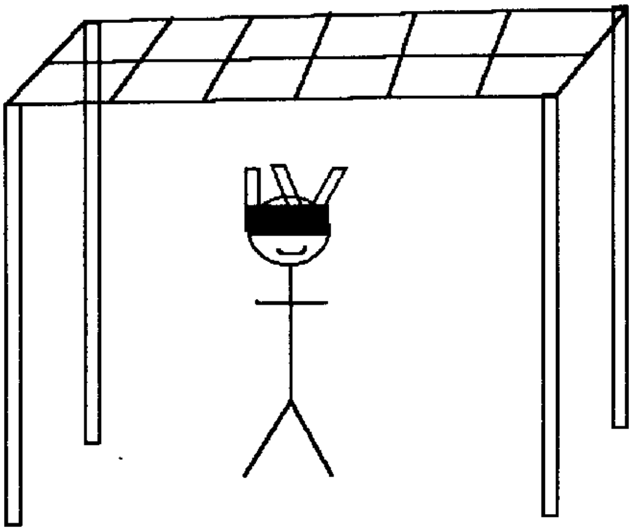


Figure 4

The system operates using a scanning method. The LED manager turns on one of the LEDs on the ceiling and records whether or not any of the sensors on the head of the user detected the LED. This sequence is continued for each of the 960 LEDs. Once this information is calculated, it is sent to the other systems for processing. The collinearity systems determines any change in detection from this set of information and the previous. Using the changes, the head position is sent to the graphics engine. The developers programmed several heuristics in the system to provide for faster head position changes.

The system suffers from several problems in LED detection. If the user moves to the edge of the ceiling panels, the system will not be able to detect the position of the head. Since the user is unaware of the edges, they will become lost. The second problem is the degree of head tilt allowed by the system. The current system is limited to +/- 45 degrees.

OUTWARD CAMERA HOMEBREW You can develop a system like the UNC system. Yes, I believe you can. You do not necessarily need a 10' by 12' area. A 3' by 3' area would suffice. Let's look at the components involved. These components (other than the computer) have not been tested.

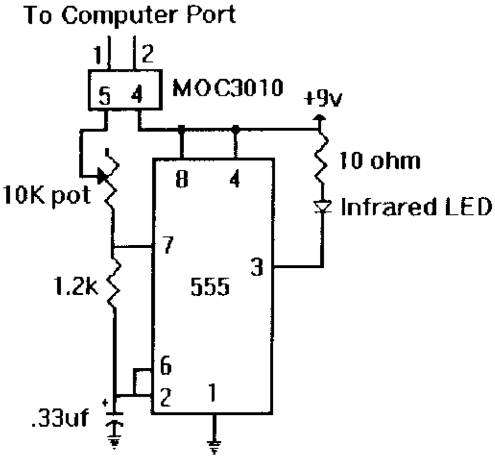


Figure 5

Infrared LED on ceiling - Figure 5 shows a simple circuit for an LED driver. Using an output from the computer described below, the LED is turned on and off.

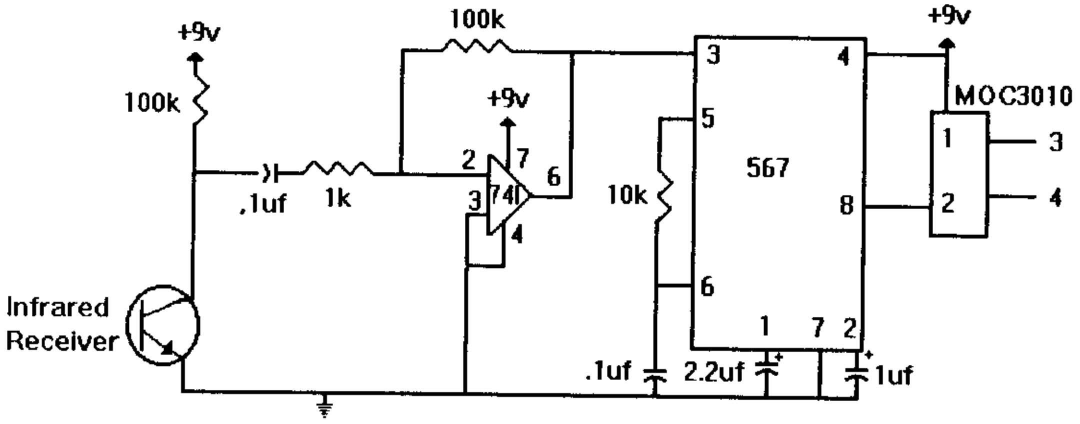


Figure 6

Infrared detection - This is a harder part to create. Figure 6 shows a simple circuit for a detector. Using a wide angle lens allows several LEDs to be detected.

Computer - The 8051/2 embedded controller from Intel can be used as an effective computer in the case of this head tracker. The bibliography includes several articles by Steve Ciarcia that document the construction of a computer using this processor. I used sixteen of these to create a parallel computer and they work great.

Software - There are many articles given in the bibliography that include the mathematics used in the UNC system.

Ultrasonic

We should all be familiar with ultrasonic positioning. The Mattel Powerglove uses this form of measure to determine the XYZ position of the glove. The same system can be used as a form of head tracking.

The Logitech 3D Tracker uses ultrasonic positioning. This is a new system that is currently available in a developer's kit for $1,000. The most common use for the system is as a 3D mouse.

ULTRASONIC HOMEBREW One potential problem with developing an ultrasonic head tracker is interference from the glove's ultrasonic mechanism. We can possibly get around this problem by picking ultrasonic frequencies different from that of the glove. There are several circuits available for a possible system. One set is based on the chips, MC14457 and MC14458. These chips were originally designed for TV remote control systems. There are 20 possible frequencies available from the transmitter. One frequency could be used for each of the three triangular points. Putting the transmitters on the head creates a cordless system.

MAGNETIC

The most common head trackers use a magnetic field to determine the location of the head. Two systems are Polhemus and an earth compass.

POLHEMUS The Polhemus head trackers are the industry standard. The Polhemus system consists of three components. A transmitting antenna, a receiving antenna, and associated electronics. Each of the antennas consists of three mutually orthogonal coils. To determine the position of the receiver, typically attached to a part of the thing being tracked, each of the three coils are independently excited. As each of the coils are excited, the associated receiver coils detect the magnetic field created from the transmitter. The controlling electronics use the resulting three vectors to determine the location of the receiver. The final information gives the computer the x, y, z, yaw, pitch, and roll of the receiver. According to current literature, Polhemus has produced three different trackers based on the above technology.

Tracker The original tracker called Tracker is currently out of production. This tracker had an effective range of 5 feet with a 40 MSEC filtered delay. The system interfaced through either an RS-232 or parallel interface. There were provisions for one to four receivers and one or two transmitters. The update rate is 60HZ for a single receiver and 15Hz for four receivers.

Isotrak The Isotrak tracker has an effective range of 3 feet with a 45 MSEC filtered delay. The only interface is RS-232. One transmitter and one receiver can be used with the system. The update rate is 58HZ.

Fastrak The Fastrak tracker is the newest tracker for Polhemus. This system has an effective range of 10 feet with an optional 20 feet cable available for larger ranges. The performance of the system decreases as the range increases. The delay is 4 MSEC unfiltered. The Fastrak does not have a filtered rate because of an advanced digital signal processing processor used for data translation. The system interfaces through RS-232, 115.2 Kbaud Max, and IEEE-488, 100kBytes/Sec Max. One transmitter can be used with one to four receivers. The update rate is 120HZ for one receiver and 30HZ with four receivers. The Fastrak has an added feature of rejecting CRT magnetic interference which neither of the other trackers provides.

COST

The final discussion involves price. The ISOTRAK system costs $3,195 which includes an RS-232 interface, one transmitter, one receiver and associated electronics.

The FASTRAK system costs $5, 750 which includes one RS-232 port, one IEEE-488 port, one transmitter, one receiver, and associated electronics.

EARTH COMPASS This systems isn't used much but could be. The bibliography includes an article on building a compass using hall effect transistors. These transistors pick up the magnetic field of the earth. Instead of lighting a simple LED, the current could be fed to an A-D converter and relayed to a computer for processing. Depending on the position of the compass, the computer would receive a different value.

To measure the degree of pan by a user, the compass could be place on top of a user's head. When they panned, the computer would be able to determine the amount. Another compass could be placed on the side of their head to measure the degree of tilt. Both of the compasses would be used for roll measurement.

There would be no head position detection from this system because we are not in control of the magnetic field. We have no way of determining the distance from the field to the compasses.

CONCLUSION

We have presented eleven different ways to track head movement. This movement can include both gaze detection and head position. In the next article, we are going to investigate the construction of a boom head tracker. I chose the boom on the sole basis that it is the cheapest to build.

BIBLIOGRAPHY

Ciarcia, Steve. "Microcontrollers" BYTE August 1988.

Ciarcia, Steve. Circuit Cellar. BYTE August 1986. pp.104.

Ferrin, Frank J. "Survey of helmet tracking technologies." SPIE Vol. 1456 Large-Screen Projection, Avionic, and Helmet-Mounted Displays (1991) pp.86-94.

Okamura, Kiyohisa, and Kamyab Aghai-Tabriz. "A Low-Cost Data-Acquisition System." BYTE February 1985. pp.199-202.

Wang, Jih-Fang, Ronald Azuma, Gary Bishop, Vernon Chi, John Eyles, Henry Fuchs. "Tracking a head-mounted display in a room-sized environment with head-mounted cameras." SPIE Proceedings Vol. 1290 Helmet-Mounted Display II (Orlando, FL. Apr 19-20 1990) pp.47-57.

Building a Boom Head Tracker

One of the simplest and cheapest forms of head tracking is a boom head tracker. Fake Space Labs makes the most popular boom head tracker. This system is part of a complete head mounted display system and not practical for our purposes.

The boom described here allows tracking of the tilting and panning of a subject's head, and allows updating as fast as the computer can read the interface. The interface used is the same circuit for the head mounted display video system that we presented last month. Add a ADC0804 analog-to-digital converter to convert the voltages to a digital form the computer can read. The overall view of the boom appears in figure 1.

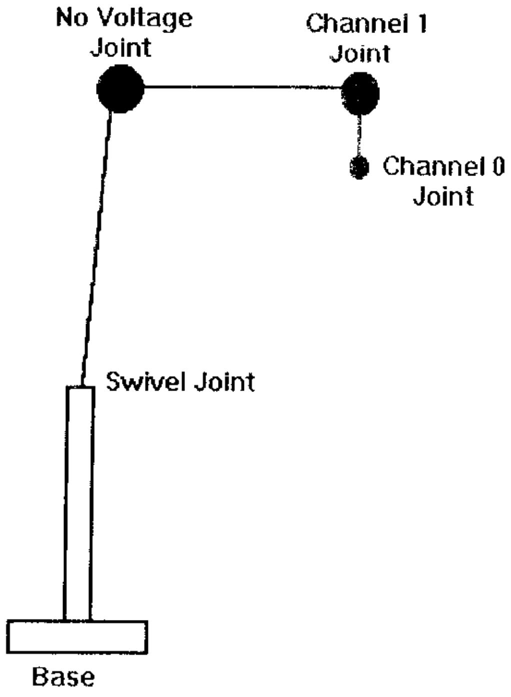


Figure 1

Basic Operation

The boom allows four degrees of freedom and measures voltage from two potentiometers to determine the location of the user's head. A third potentiometer can be added for a 6 DOF system. Using Figure 1 as an illustration, the boom can be described. The Channel 1 Joint measures the degree of tilt the user's head is experiencing. The Channel 0 Joint measures the degree of pan the user's head is experiencing. The No Voltage Joint and the Swivel Joint allow the boom to flex when the user moves their head. Attached to each Channel Joint is a 1 Megaohm potentiometer which is fed a +5V voltage. As the user moves his head in a particular motion, the degree of voltage through the potentiometer changes. If the voltage of the potentiometers is known when the user is looking straight ahead, then any decrease or increase in the voltage indicates a head movement. The analog-to-digital converter translates the voltages into digital form for the computer.

Boom Construction

The construction of the boom varies from extremely inexpensive to expensive. The prototype version created by the author was constructed for under $10.00 including the cost of the electronics. Each part of the construction of a low-cost head tracker is examined.

Base The boom must be constructed in a sturdy fashion so that the user does not feel the interaction of the boom. The base of the boom should be approximately the height of the waist of the user. This requires that the base be adjustable. One of the simplest bases is a camera tripod. Alternatively, a base can be build out of several 2x4 boards with an adjustable middle board for different heights. Figure 2 shows an example.

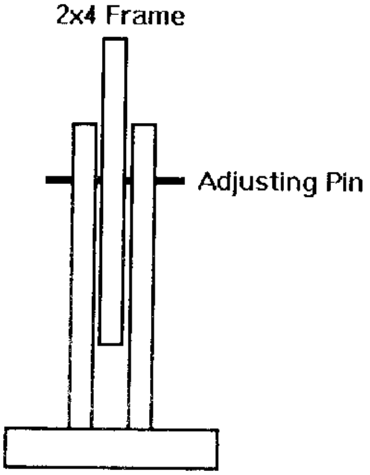


Figure 2

Drill holes in the three pieces of vertical 2x4 wood at regular intervals. When an adjustment is necessary, remove the pin and move the middle 2 x 4 up or down and replace the pin. This base can be built for approximately $3.00.

Boom Arms The boom in figure 1 has three arm pieces. The arm attached to the base of the boom was constructed out of 1/2" wood dowel. This dowel is attached to the base using a small hinge. Figure 3 shows an example. The user of the boom is placed with their back to the dowel. For a stronger arm, the wooden dowel can be replaced by aluminum mast poles or a 3/4" dowel can be used.

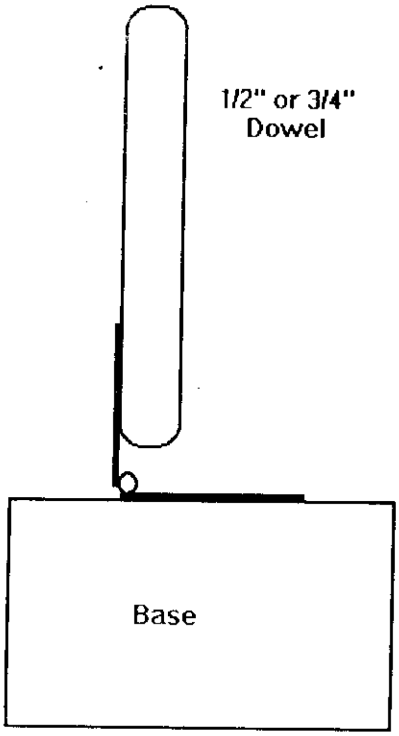


Figure 3

The second arm is attached to the base dowel and the shorter dowel at the end of the boom. These arms are made from 1/4 dowels. The main reason for using the dowels in the construction of the boom is lightness. Steel rods are too heavy and the aluminum mast poles are considerably more expensive.

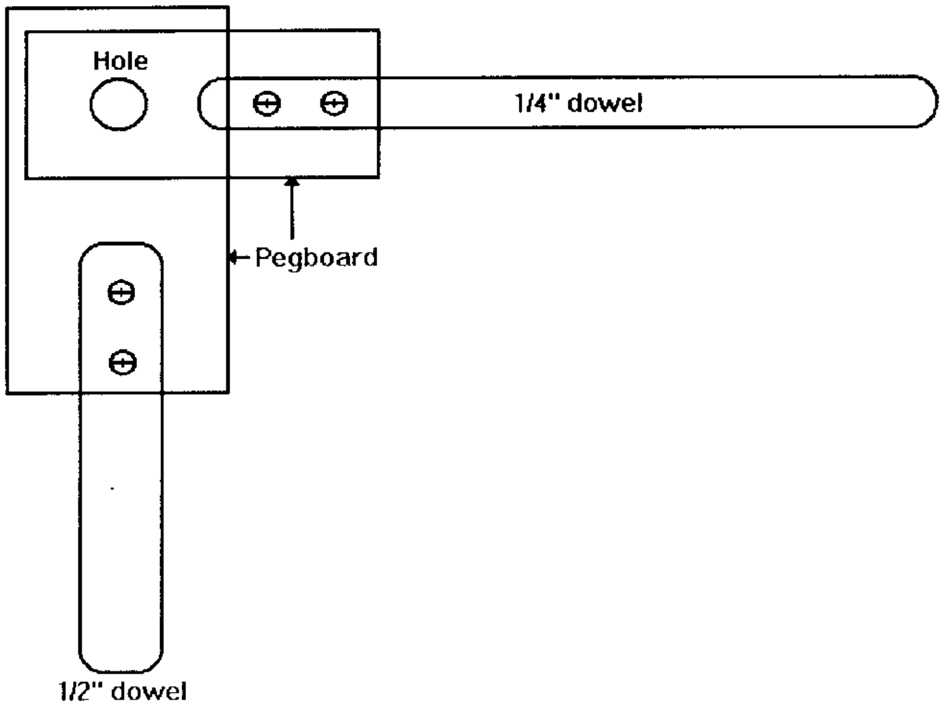


Figure 4

Boom Joints Figure 4 shows the No Voltage Joint between the base dowel and the second arm. This joint should be moveable. Each of the dowels has a small piece of pegboard or other 1/8" piece of wood attached to it by two small screws. The pegboard should be mounted on opposite sides of the two different dowels; one on the front of one dowel and the other on the back of the other dowel. This is done so when the pegboard are bolted together, they are free to move without hitting the dowels themselves. In the upper part of each pegboard is a hole cut slightly larger than a 1/4" bolt. The two pegboard pieces are put together using a 1/4" bolt and a two nuts. The second nut is used to lock the first nut in place. The bolt should be loose enough to allow the two dowel to act as an elbow joint. When this section of the boom is mounted to the base, the end of the second arm should be free to move forward and backward with little interference.

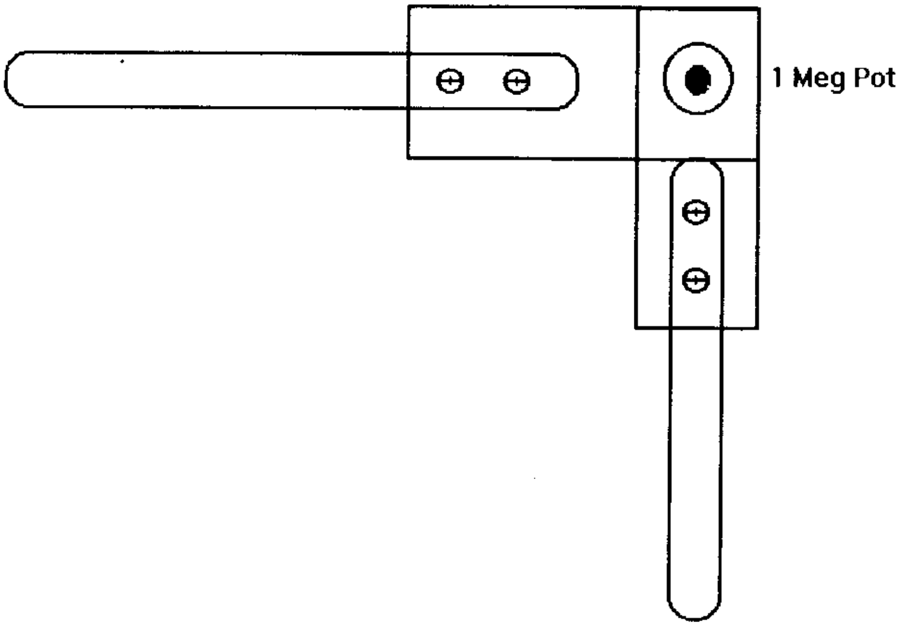


Figure 5

Figure 5 shows the Channel 1 Joint for the boom. This joint measures the degree of tilt in the user's head movement. As in the No Voltage Joint, two pieces of pegboard are attached on opposite sides to two different 1/4" dowels. In the upper section of one of the pieces of pegboard, a hole is drilled large enough for a 1 Megaohm potentiometer. Radio Shack has one for $1.19. This potentiometer should be firmly attached using the provided washer and nut. In the upper section of the other pegboard, a hole is drilled just slightly smaller than the shaft of the potentiometer. This pegboard is slid over the shaft of the potentiometer. Before the pegboard is attached to the shaft, the potentiometer should be rotated to the middle of its rotation. The quickest way to determine this point is to use the calibration method described later in this article. Once the middle point has been found, the pegboard should be attached to the shaft with the shorter dowel vertically placed as in figure 5. The system can be tested at this point by hooking up the potentiometer to the ADC circuit and executing the ADC.PAS program found on the enclosed disk. Place the end of the short vertical dowel straight in your hand. Now move your hand in a forward tilting motion. The numbers for this potentiometer should change. Now return your hand to the vertical position. The numbers should be very close to the middle values calibrated above. Now move your hand in a backward tilting motion. The numbers will change in the opposite direction. The computer will use these numbers to determine the user's head tilting.

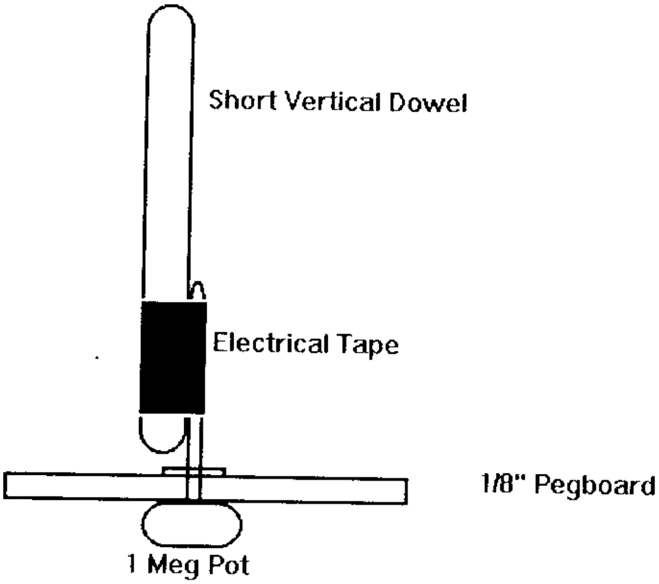


Figure 6

The last joint for the boom is used to measure the degree of panning by the user. Figure 6 shows the construction of the last joint. A 1 Megaohm potentiometer is mounted to a small piece of pegboard. This pegboard is in turn mounted to the top of the user's head in some fashion. Some options are to use an elastic band or attach it to a head mounted display casing. The shaft of the potentiometer is attached to the small vertical arm using duct or electrical tape. This potentiometer can be tested in the same way that the previous one was.

6 DOF

To provide for measurement of the roll of a user's head, a third potentiometer can be attached to the system. This potentiometer is described now because it did not work to my satisfaction when included in the main design. In figure 4, instead of simply attaching the second arm to the No Voltage Joint, a potentiometer can be used as a jointing mechanism. Figure 7 shows the design.

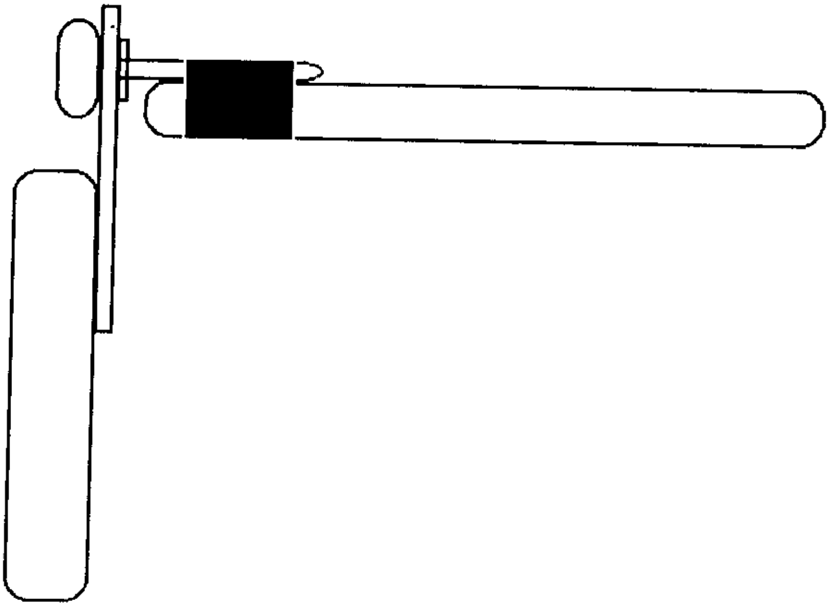


Figure 7

As the user rolls their head from side to side, the potentiometer measures the change in voltage. The problem with this design, torque, is created when the user moves both in a roll and tilt or roll and pan motion at the same time. A cheap solution to the problem is currently being worked on.

Hooking Up The Potentiometers

After the construction of the boom apparatus is complete, the potentiometers have to be hooked up. The current design of the boom uses two channels of an eight channel analog-to-digital circuit. In addition to the two channels, a voltage source is needed for a total of three wires. Each of the potentiometers have three connections, only two need to be used. One wire should be designated as the voltage source and should be so marked. Attach this wire to the far left connector of both potentiometers when looking at them from the back (shaft pointing away from you). Attach a second and different wire to the middle connector of the potentiometer at the very end of the boom. This is channel 0 and should be so designated. Attach a third and different wire to the remaining potentiometer. This is channel 1 and should be so designated.

ADC Circuit

Figure 8 shows the schematic for the analog-to-digital circuit used for the boom. There is a total of eight channels available to use but only two will be used for this boom, possibly three for the optional potentiometer.

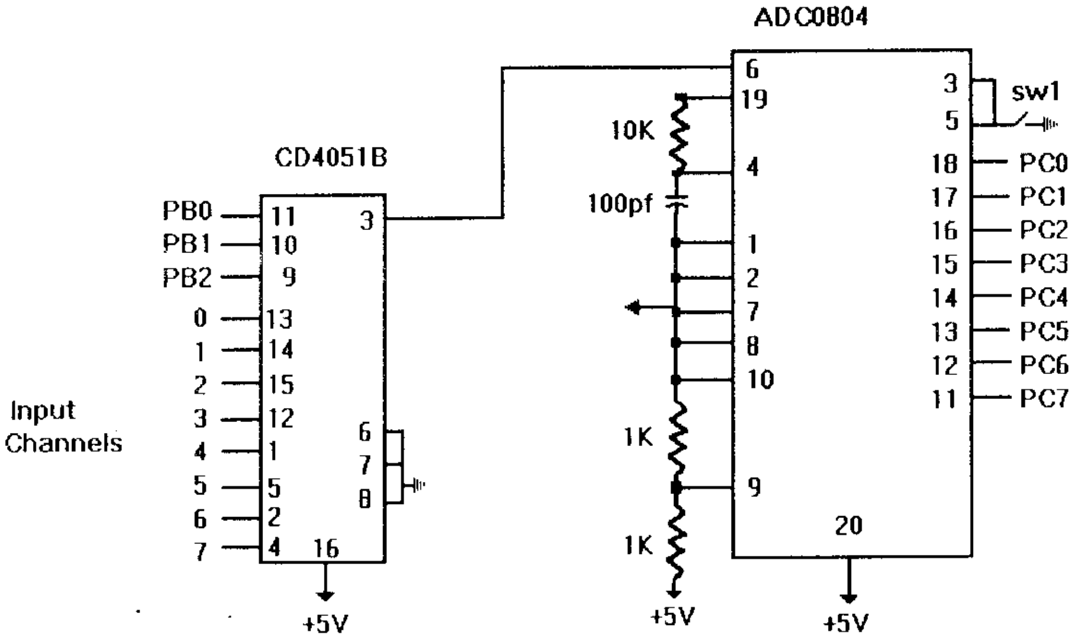


Figure 8

The circuit is fairly simple to build. The eight data lines on the ADC chip are connected to the C port of the 8255 interface presented last month. The three address lines on the CD4051 chip are connected to the first three lines of the B port. Each of the input lines on the CD4051 are selected by setting the correct PB0-PB2 line high. If you wish to measure the voltage on line 4, set PB0-PB2 to the value of 4. The ADC chip would take readings from that line.

Switch sw1 resets the ADC. When the circuit is first powered up, this switch should be tripped to allow the ADC to reset itself.

Calibration

In order to determine the range of a potentiometer, we need to calibrate it. The program ADC.pas is a short program which shows how to use the ADC circuit. This program will read voltages from channels 0 through 2 and report them on the screen until a key is pressed. When a key is pressed, the highest and lowest values for each of the channels is shown. These values can be used to determine the middle position of the potentiometers. When attaching the arms of the boom to each other, use this program to move the potentiometer to its middle position.

Software System

Programming the ADC circuit is not complicated. It is assumed that you have built the interface circuit and the circuit above. The interface circuit will reside at I/O location 640 through 643. The first step in using the circuit is to tell the interface that Ports A and B are output and Port C is input. This is done with the instruction (in C)

```
outportb ( 643, 137 );
```

After you tell the interface the correct port setup, you instruct the CD4051 chip to select an input line with the instruction:

```
outportb ( 641, x );
```

The value for x depends on the input line number. In this case, you just use the actual number of the input line we wish to use. After this step, you simply read the ADC chip with the instruction:

```
value = inportb ( 642 );
```

If you set up a loop using the above instruction, you will find that the values read from the ADC jump around. To solve this problem, you introduce a short delay into the read. Before obtaining the value with the instruction above, do a delay of 10 milliseconds:

```
delay ( 10 );
```

This gives the ADC time to produce a correct answer for an input voltage. This becomes especially important when using more that one channel. The following sequence would be read values from two different input channels:

```
outportb ( 641, 0 );
delay ( 10 );
```

```
value1 = inportb ( 642 );
outportb ( 641, 1 );
delay ( 10 );
value2 = inportb ( 641 );
```

The delays are important because the ADC will be receiving a different voltage from the new channel. This gives the ADC time to generate the correct value.

Rend386 Code

After using the ADC.pas program for a time, you may feel the need to introduce this head tracker to Rend386. The program headt.c is a simple program that reads in a figure file and enables use of the head tracker. You can test the head tracker by running this program and the demo3.exe program to see that the keyboard head movements do indeed match those of your new head tracker.

The headt.c program determines head movement in three steps:

1) It determines the position of the potentiometers for a persons looking straight ahead. The code that accomplishes this is:

```
printf( "Stand with your hand in an upright forward looking position\n" );
printf( "While I find the position of your head. (approx. 6 seconds)\n" );
delay ( 5000 );

outportb ( 641, 0 );
delay ( 10 );
panmiddle = inportb ( 642 );
pancurrent = panmiddle;

outportb ( 641, 1 );
delay ( 10 );
tiltmiddle = inportb ( 642 );
tiltcurrent = tiltmiddle;

printf( "Thank You!" );
```

The current position of the head for both the tilt and pan motions are kept in the variables **tiltmiddle** and **panmiddle**. These are used as reference points. The current position of the head will be kept in the variables **tiltcurrent** and **pancurrent**.

2) After we have the starting position, we can begin to read the position of the head in a regular fashion. We introduce code into the main loop of all Rend386 programs:

```
outportb ( 641,1 );
delay (10);
value = inportb(642);
```

```
if ( abs(value-tiltcurrent) > 5 ) process_tilt(value);

outportb ( 641,0 );
delay (10);
value = inportb(642);
if ( abs(value-pancurrent) > 5 ) process_pan(value);
```

This code sets the correct input line, delays, and read the port. A test is made of the new value to determine if an acceptable amount of head movement has occurred. If the new position is more than 5 units in either direction, process the new position otherwise, throw outt the new read. This is done for both potentiometer. If there was acceptable movement, call one of two functions; **process_tilt** or **process_pan**.

3) Each of the functions moves the eye position in Rend386 according to the value it is given as a parameter. The **process_pan** function is:

```
void process_pan ( unsigned char value )
{
  latitude = (value-panmiddle)*2*65536L;
  polar_compute();
  compute_view_factors ( current_view );
  pancurrent = value;
  redraw = 1;
}
```

The original demo programs for Rend386 included a function called polar_compute. This function used latitude and longitude data to determine a new eye position based on the polar coordinate system. Use part of the polar_compute function to create the new eye position. For the pan motion, the global variable **latitude** is set to twice the angle generated by the current head pan position and the earlier recorded pan middle position. The function **polar_compute** is called to set the new eye position and the function **compute_view_factors** is called to do the Rend386 housecleaning for view positions. Record the new current position then indicate that the screen needs to be redrawn.

The same type of function is used for the tilting motion of the head except that a longitude global variable is used in the polar calculations.

Conclusion

I have presented a simple boom head tracker that can be build for under $10. This head tracker works correctly and provides a new level of Virtual Reality for the experimenter.

Virtual Imagination

Virtual Imagination is a new column that will cover the pondering of two virtual dreamers. The authors are:

Shawn Casey B.S. in Finance with a minor in Computer Science
Steven Casey B.A. in Management Information Systems

We are the "two friends" who introduced your publisher to virtual reality. It was a late night jam session at our apartment, well that's another story. But, a month or so later he told us that he was planning to publish a journal on virtual reality. We thought he was crazy! And now we are writing for him, well stranger things happen.

We can't give you an accurate definition of what topics that we will cover in the column. We feel that once you limit the areas that will be covered the content becomes very static. A lot of our ideas will come from the news media or the "experts" in the current VR revolution.

What Virtual Imagination is?

Plato pondering the meaning of life, while wired to a 486 (50 MHz) personal computer, full color HMD and a hot wired Nintendo PowerGlove.

What it isn't! Techno-babble such as:

- Is a PC fast enough to do interactive 3D graphics?
- Which University has the best VR research group?
- Is virtual sex morally acceptable?

Virtual Imagination is limited to the bounds of our imaginations and the tolerance of our editor. We have a slight understanding of electronics and computers, but, don't claim to be experts. Therefore, virtual imagination will be limited to fictional ideas and the exploration of our imaginations about the future (past and present) of VR. We would like to evoke controversy. Please write to us, we will read your letters. Maybe your imagination is running wild with a VR idea. We would like to hear about it! Send your letters to:

PCVR
Attention: Virtual Imagination.
1706 Sherman Hill Rd. #A
Laramie, Wyoming 82070

Dream A Life

I saw a movie the other day, it wasn't a very good movie, but, it did cover some interesting topics. The movie was Megaville, I know, what should I expect from a movie with a title like that? But, the display box (VHS tape box) looked rather interesting. It stated that this was the near future of our world, a world with widespread mind control. Those are two topics that always get my attention, and for $0.50 rental fee, how could I pass it up. Well the movie is pretty stupid and I'm being kind. It had a plot that skipped and was hard to follow or even stay awake through. Now that I have ripped the movie apart, you must be wondering what is Dream A Life? The interesting topic in this movie was the new banned media of the future or Dream A Life.

Dream A Life was a small electrical device that rested across the front of your head like a sweat band. It emitted low current electrical stimulation to the brain to induce hallucinations. These hallucinations were controlled to the point that the user could choose from different fantasies. Are you starting to get the picture? We are talking about VR without the HMDs and the computers.

IS IT POSSIBLE?

I don't know if Dream A Life is possible, but, I do know that it is possible to "create" hallucinations. I have been reading a book, you guys will love the title, MegaBrain by Michael Hutchison. I don't understand, but, it seems like the new catch words for brain expansion/stimulation must involve the word "mega". But, back to the point, in this book the author explains the research involving brain stimulation. Researchers have devices that can change the emotions of the users by low level electronic stimulation of the brain. They have found currents and wave forms that cause pleasure, nausea, fear, etc. Anyone interested in this topic should read the book, because I can't do it justice. As a matter of fact, take a few of the dollars that you use on video rentals and buy the book. It is much more stimulating than most of the "new releases".

IS IT POSSIBLE TO CONTROL HALLUCINATIONS?

I have no idea. Ten years ago, if you would have told me about VR, I wouldn't believe it either. So, who knows, maybe NASA? If it really could exist, think of all the benefits. It could be the future of VR. You wouldn't need HMDs or Head Trackers. We might even have a quick enough processor between our ears to offer real time VR!

HOW COULD THIS TECHNOLOGY BE USED NOW?

I believe that the mind expansion/stimulation devices could be integrated with VR. Think about it, we are talking about computer generated emotions. Not just sight, sound, smell, etc., but now the technology exists to control emotions. Video games that could screw up your mind. We don't need "killer graphics", we could just scare the heck out of the consumers. I know, I'm back on the soap box. But, this is a very interesting area. You will notice that I didn't mention morals or the legality. This area could be exploited by unscrupulous business people to make VR games addictive. But, on the other side brain explorers (mad scientist) could generate computer emotions. With a simple D/A circuit and a device described in Hutchison's book, VR explorers could add a new sense to their virtual world. I'm not trying to make this sound too simple, I know that there are a few details that I have ignored. Such as, the machines are only available to licensed practitioners and are quite expensive. But, the whole point that I am trying to make is that it is possible.

If anyone wants to try it out, I would enjoy hearing about your findings. The movie (Megaville) tried to bring out the bad things with this new technology (Dream A Life). Such as, its ability to be used as a mind control device. I think that is what is wrong with the majority of our society. They look for the bad in technology, without looking at the good. Think about all of the politicians and business people who would love to be able to control your emotions! Tongue in cheek.

Back to VR explorers, this is a new sense that we can add to our computer worlds. And, used carefully, it could imitate emotions that occur naturally in our real world. Well that's all that I could pry out of my virtual imagination for this issue. In closing, Who wants to help me make Ms. Pacman physically addictive?

Building a Head Mounted Display

This article deals with the optics in our series on building a head mounted display. For the homebrewist, the optics are probably the most difficult part in constructing a head mounted display because of the cost. My experimentation was limited and I am fully aware that there are better optics than those I present. However, the optics we build in this article do work and are not expensive. I assume for this project that the LCD's used in your head mounted display are those found in on-the-shelf televisions.

Background

One of the first questions that must be answered is why do we need optics? Why can't we just put the LCDs in front of our eyes and see 3D? In our case, the answer to these questions are two-fold.

First, if you take two of the Casio 2.2" televisions are try to put them in front of your eyes, you will find that the left eye looks at the right part of one of the TVs and the right eye looks at the left part of the other TV. We ideally like to look at the middle of each screen. We could compensate for the difference in viewing area but this significantly reduces the field of view of the user.

The second reason is the focusing distance of the eye. To create a true three-dimensional image, a left image must be presented to the left eye only and a right image must be presented to the right eye only. If the two screens are put in front of the eyes at the distance that they can actually focus on the screens, the eyes will operate as they should by focusing on a single object. We need the eyes to focus separately on each of the screens.

If we place the screens close enough to the eyes to separate them, we cannot focus on the image on the screens. To relieve this situation, we introduce optics. Figure 1 shows a reproduction of an early stereoscopic apparatus taken from the book "THE PRINCIPLES OF OPTICS" by Hardy and Perrin.

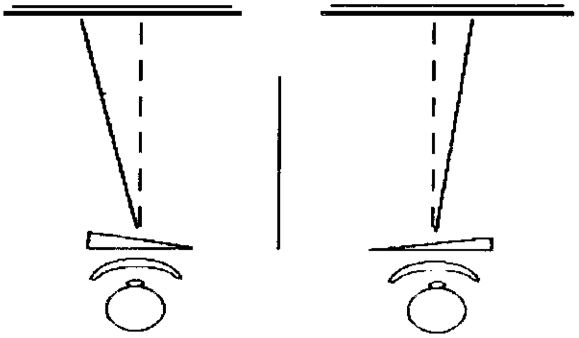


Figure 1

Each of the eyes view parallel paths to the separate screens. The large black horizontal marks are prisms that allow for larger pictures to be viewed. The lens at the bottom of the illustration allow the user's eyes to focus correctly on the separate images. The line in between the eyes guarantee that each of the images remain separate. The idea of stereoscopy has been around for a long time. The book that this illustration appeared in dated back to 1932.

The optics for head mounted displays is primarily controlled by a company called LEEP Systems. They produced the LEEP optical system. These optics consist of three different lenses per eye. For a thorough description of the LEEP Optics look at the article "Wide Angle Orthostereo" by Eric M. Howlett in SPIE Vol.1256 Stereoscopic Displays and Applications (1990) and the article "A Computational Model for the Stereoscopic Optics of a Head-Mounted Display" by Watten Robinett and Jannick P. Rolland in SPIE Vol.1457 Stereoscopic Displays and Applications II (1991).

The LEEP optics have a 140 degrees field of view if an image is presented in the correct format for the optics. A field of view is the amount of visual space the eye has for viewing objects. A normal eye has a field of view of 270 degrees. If you look straight ahead, you are able to see movement up to 90 degrees to the right or left. This is a total of 180 degrees. The eye is able to move 45 degrees in either direction, so if you look all the way to the right and then sweep all the way to the left, you are able to see about 270 degrees. If you add the fact that we can also turn our head around, we can really see a total of 360 degrees. For this head mounted display, you should create the total 270 degree field of view.

The only problem is creating a software system that will display fuzzy images for our peripheral vision and creating a LCD screen that is 270 degrees around. Our only solution is to settle for a smaller field of view.

Our Optics

We begin the construction of our optics by moving the LCD image to the middle of our eyes. This involves spreading the small televisions far enough apart to handle the power connection. In doing so, you will notice that both eyes focus on the edge of the televisions. To move the video image to our eyes, we use an old trick we learned as a child.

A periscope allows looking around a corner or above something. A simple periscope moves the video from the LCDs to our eyes. Figure 2 looks down at the inside of one of the periscopes.

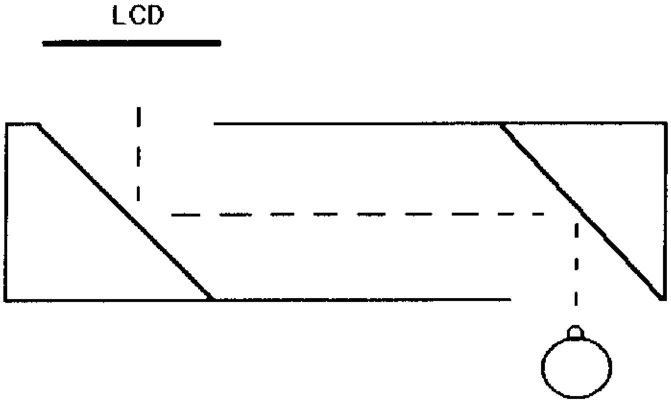


Figure 2

Careful construction of the periscope is vital to making the HMD work correctly. I will talk about how I made my periscopes to give you some idea on how to make one.

Using a large piece of black poster board, I cut out a five sided box. The sides of this box are the same length as the height of the LCD screens. The box was assembled with one side overlapping another for strength. Electrical tape holds the box together.

View Openings

The next step is to create an opening for the LCD screen. You also need an opening where your eye peers through. These openings should be on opposite ends and sides of the box just constructed. Cut the opening for the LCD screen approximately 1/4" from the end of the box as Figure 2 shows. This opening should be large enough to allow all of the screen to appear. The eye opening should be cut to the height of the box and approximately 3/4" wide.

Mirrors

A periscope operates by bouncing an image off of two mirrors positioned as in Figure 2. You can purchase a small mirror at Walmart for $0.97 and a mirror cutter for $4.00. This mirror gives you enough room to cut the four necessary mirror segments and several mistakes. Cut two segments as tall and slightly longer than your LCD screens. Position the mirrors in the box as shown in Figure 2. The mounting of the mirrors can be accomplished using two small slits in the box or using electrical tape. The following are two important things to watch out for when placing the mirrors:

1) Make sure that the mirrors are perfectly vertical. If either of the mirrors lean to one side or another, your image will appear to be crooked.

2) The mirrors must be at parallel angles to one another.

The easiest way to determine if you have the mirrors positioned correctly is to simply use the box as a periscope. If you hold the box perfectly horizontal, you should see an image of the screen to your right if you are using the right eye. The image also should be perfectly straight. Adjust the mirrors if the image appears to lean or you see other parts of your environment.

Lenses

The last part of the construction of the optics is to install lenses. The lenses we use can be purchased at Walmart. They are full sheet page magnifiers. They cost $1.00. You will need a minimum of two. If you look at each of the sheets, you see a small dot in the middle with many circles emanating from this dot. Cut a square piece of the lens with the dot in the middle slightly larger than your LCD. Mount this lens on the front of the periscope just built. The dot should be in the middle of the LCD cutout opening. The purpose of the lens is to allow you to focus on the LCD screens at a reasonable distance.

End

Once you have one of the periscope optics finished, you should build the second one. Both of the periscopes have to be attached to the LCD screens. In the case of the Casio televisions, there is enough plastic border around the screens to allow for a second small box to be constructed over the lens and the LCD. The optics should be placed approximately one inch from the LCD screen. Experiment with this distance to determine the exact measurement for your eye to focus on the screen.

Enhancements

The above setup gives you a horizontal FOV of about 75 to 80 degrees. You can extend this FOV by introducing a second lens. Figure 3 shows the placement of the second lens.

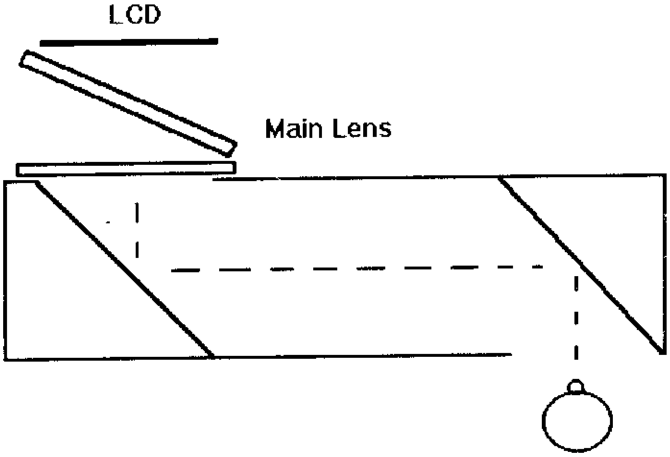


Figure 3

The second lens causes the image on the LCD to widen. This increases the FOV by 10 to 20 degrees. However there is a down-side. The image becomes smudgy and appears somewhat out of focus. You can try adjusting the distance between the lenses and the LCD to combat this problem. But it cannot be totally eliminated. An additional enhancement to the system is the use of right-angle prisms. These prisms bend light 90 degrees. The prisms can be used in place of the mirrors for a slightly better image transfer. The prisms are a little costly, approximately $25.00 each.

Conclusion

The ideal optic system for head mounted displays is the LEEP optics. However, using simple materials, an adequate optic system can be created which provides the necessary level of immersion for the user.

Graphics

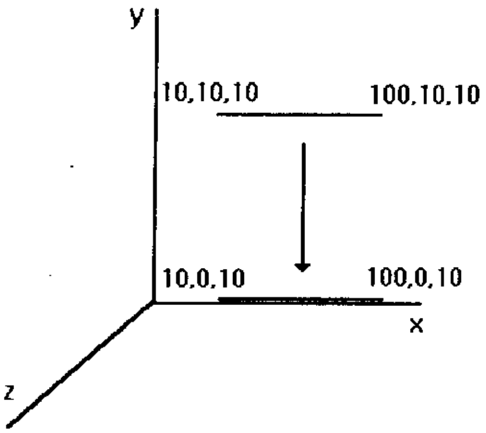
In the last installment of the **GRAPHICS** column, we discussed rotating a three dimensional object about a coordinate axis. This type of rotation is not useful for most graphics projects. Rotation about a particular line is better because we can rotate an object in any direction we wish.

The program axis4a.c on the enclosed disk shows the problem. The blue line is defined as 10,10,10 to 100,10,10. This line is not associated with any of the coordinate axes. By pressing the X key, the line rotates about the X coordinate axis. Notice how the line moves out and away from the viewer as the line continues to rotate. The desired action is to have the line simply rotate where its located. Since we are dealing with a line, there should be no visible change to the line itself.

If the object that we wish to rotate is parallel to the coordinate axis of the rotation, we can make three simple translations to achieve the desired rotation.

Translate To Coordinate Axis

The first step is to translate the object to the coordinate axis. Figure 1 illustrates the movement.



Since we plan to rotate the line about the X coordinate axis, we move the line to that axis. This is done by translating the line on the Y coordinate. The translation matrix is:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -10 & 0 & 0 \end{bmatrix}$$

Once this matrix is applied to the line, the line will be on the X coordinate axis.

We do not want to redraw the screen at this time because we have not completed all of the rotation.

Rotation

The next step is to perform the desired rotation. Recall from the previous column how a rotation is performed about the X coordinate axis. The matrix is:

$$\begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\cos\theta} & \boldsymbol{\sin\theta} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{-\sin\theta} & \boldsymbol{\cos\theta} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix}$$

Using the desired angle of rotation, this matrix is applied to all the points of the object. The object now rotates in the coordinate fashion. If we redraw the screen now, we would have the correct rotation but the object would be out of place.

Return Translation

The third step is to return the object to its original position. This is performed using the same translation matrix given above except the translation values are reversed. If the first translation had positive values, the return translation would have negative values. The same goes for the reverse.

In total, we have three matrix operations that must be performed on the object. One way to simplify the calculations is to combine all three matrices into one.

Transformation Matrices

The combined result of the individual matrices is called a **transformation matrix**. These matrices have any number of rotation, scaling, and translation matrix combinations. The most important part in creating transformation matrices is the order of the combination. In the above rotation example, we performed the following matrix operations:

- Translation to coordinate axis
- Rotation
- Return translation

When these operations combine into a transformation matrix, the results of the application of the transformation matrix must be the same as the application of each individual matrix.

Building Transformation Matrices

To build a transformation matrix, we lay out the required individual matrix from left to right. The left-most matrix is the first individual matrix to apply to an object. The right-most matrix is applied last. In our case, we have:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -10 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{\cos\theta} & \boldsymbol{\sin\theta} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{-\sin\theta} & \boldsymbol{\cos\theta} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{1} \end{bmatrix} \quad \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -10 & 0 & 0 \end{bmatrix}$$

Each of the matrices lined up are multiplied to each other starting on the left and working towards the right. The final matrix is the transformation matrix. In our case, the first multiplication generates the matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & -10\cos\theta & -10\sin\theta & 1 \end{bmatrix}$$

The second and final multiplication generates the following transformation matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & \sin\theta & 0 \\ 0 & -\sin\theta & \cos\theta & 0 \\ 0 & -10\cos\theta+10 & -10\sin\theta+10 & 1 \end{bmatrix}$$

This transformation matrix can be used in place of the three individual matrix operations above. This may seem like extra work but it pays off in calculation speed.

Computations

As an example, if we perform the above rotation on an object that had 25 vertices, we would need the following number of matrix operations:

* 25 - translations to coordinate axis for a total of 75 multiplications
* 25 - rotations for a total of 75 multiplications
* 25 - return translations for a total of 75 multiplications

* A total of 225 multiplications

Using the transformation matrix we have:

* 2 matrix multiplications for a total of 32 multiplications
* 25 - transformation uses for a total of 75 multiplications

* A total of 107 multiplications

The transformation matrix decreases the number of multiplications for this rotation by 118, which is a significant reduction.

Program

The program on the enclosed disk called axis4b.c shows the new rotation in both individual and transformation styles.

Working With Rend386

Welcome to the first installment of **Working With Rend386**. This column is intended to give you working knowledge of Rend386 as we build an application together. The application I have chosen is a racquetball game. Over the next few issues I will discuss topics such as:

* Building a racquetball court
* Making a ball bounce
* Hitting the ball with the racquet

Each of these topics touch on the very foundation of Rend386 and explore new issues such as collision detection and real-world movement of objects. This first column talks about building a racquetball court.

Rend386 Release 4

The new release of Rend386 should be available by the time you receive this issue. In the meantime, I am going to show you how to develop a court using the demo program. Since the only interaction needed for the court is eye movement, this is an acceptable foundation.

The new version of Rend386 can be used to develop worlds. These worlds can be build using 'splits'. A split is a mathematical plane that divides space into two different areas. There is space on one side of the split and space on the other side. Splits are defined by specifying a point in the plane and by a normal to the plane or by specifying three points in the plane. Each of the walls of the racquetball court will be defined as a split. Any lines on the courts floor or walls will be 'on' the splits.

Let's begin by creating a very simple world. The world.doc file included with the demo release describes the commands that can be used to create worlds. Some of these commands will be included in all worlds.

worldscale value - This command specifies the scale of the world as a floating point number. The **value** is given as millimeters per unit.

start x, y, z pan, tilt, roll zoom - This command gives the starting location of the eye. The x, y, and z values are long integers. Pan, tilt, and roll are floating-point angles. Zoom is a floating point number. Notice that there are NO commas between the Z and PAN and between the ROLL and ZOOM values.

hither value - This is the near clipping distance given as a long integer in world coordinates. The documentation recommends a value or 10 or more.

yon value - This is the far clipping distance given as a long integer in world coordinates. The documentation recommends a value of 1000000 or more.

Once these main commands are set up in the world file, the surfaces and colors of our objects need to be defined. This is done using three commands: **surfacedef**, **surfacemap**, and **surface**.

Surfacedef - This command associates a name with a color value. The documentation file **colors.doc** describes the format of color values in Rend386. This simply gives a name to a specific color. An example from the room4.wld file is:

```
surfacedef marble 0x1DF8
```

Surfacemap - This command is a grouping function. A group name typically follows the surfacemap command and is followed by any number of SURFACE commands. The purpose is to associate colors to objects. An example from the room4.wld file is:

```
surfacemap rugcolors
surface 1 beige
surface 2 blue
surface 3 red
surface 4 orange
```

Surface - This command is used with the surfacemap to define surfaces. The command is followed by a surface number and a surfacedef name. In the example above, we are saying that surface number 1 in the group rugcolors is the color beige. The surface numbers are used in the .plg files. A simple .plg object would be:

```
junk 4 1
0 0 0
0 10 0
10 10 0
10 0 0
0x8001 0 1 2 3
```

The object is a simple plane with a color of 0x8001. The color documentation indicates that if the high bit of the first byte is set, the least significant 15 bits are an index into a surfacemap. When this object is used in a world, it will be painted the first color in the specified surfacemap.

OBJECTS, SPLITS, and POLYOBJ

Those who have started to use splitting planes have probably found that the documentation provided in the demo program lacks any kind of example. Trial and error becomes the best method of using splitting planes. Let's give it a try

Recall that a splitting plane is a mathematical plane that separates space into two different parts. The parts define the leaves of a binary tree. If you had three splits defined as follows:

```
          area5             2            area1
                            2
  111111111111111111+11111111111111111111
                            2
        area2               2            area3
                            2
   33333333333333333+3333333333333333333333
                            2
        area6               2             area4
```

This would build a binary tree. The first split would give you a tree

```
        Split1
           /\
          / \
      area1  area3
```

with area2 and area4-area6 not visible. If you add the second split, the tree becomes:

```
        Split1
           /\
          / \
    Split2   Split2
      /\        /\
     / \       / \
    a1  a3    a5  a2
```

The third split produces the following tree:

```
          Split1
             /\
            / \
     Split3    Split3
       /\          /\
      / \         /  \
      /  \       /    \
  Split2   a4 Split2  a6
    /\          /\
   / \         / \
  a1  a3      a5  a2
```

Now if you put an object in area1 and the same object in area4, the object in area1 should be totally displaced by the object in area4. Following the tree gives us that result. We travel down the left node of Split1, left node of Split3, left node of Split2 and render any objects in area1, then back up the tree to the right node of Split2 and render any objects in area3. You travel back up to Split3 and down the right edge to render any objects in area4. Since our objects are the same but in different areas, the object in area1 will be displaced by the object in area4.

Defining Splits

Split are defined using the world file command SPLIT x, y, z nx, ny, nz flags or SPLITSPTS x1, y1, z1 x2, y2, z2 x3, y3, z3 flags. The first command takes any point on the plane of the split and a normal to the desired plane. A normal is a line that forms a right angle with the desired plane. As you read this article, the path from your eye to the paper is a normal vector. To define a wall that is parallel to the z coordinate plane, the normal would be 1, 0, 0. There is no documentation on the purpose of the flag value.

Our racquetball court will have one split in it. This split will represent the back wall of the court. To create the wall, we have to define the split. The split is:

```
split 800, 1, 4000 0, 0, 1  1
```

The wall is created from a .plg file copied from the room4.wld world.

```
wall 4 2
0 0 0
0 50 0
100 50 0
100 0 0
0x8001 4 3 2 1 0
0x8001 4 0 1 2 3
```

Each side of the wall is colored using the first surface value defined for a specified surfacemap. The actual wall is put into the world using the **object** world command.

The **object** world command has the following format:

OBJECT .plgfilename sx, sy, sz rx, ry, rz tx, ty, tz depth mapping parent

The sx, sy, and sz represent the scaling functions for the object. The rx, ry, and rz represent the rotation functions for the object. The tx, ty, and tz represent the translate functions for the object. Depth is the type of depth sorting to use on this object. Mapping is the surfacemap group name to use for coloring this object. Parent is the name of the object that this object is a child of. If the object has no parent, this field is left blank. If the object is to remain in a fixed position, the word FIXED should be used in this field.

The back wall of the racquetball court is created using the command:

```
object courtwall  32, 48, 4  0,0,0  800,0,4000 0 walls fixed
```

Any number of splits can be used in a world. Once all splits have been defined, the command ENDSPLITS must appear in the world file. Objects in the areas of the splits do not have to be created when the split is defined. However, this is an easy format for defining walls. The splits for the ceiling and floor are:

```
split  0,2400,0 0,0,0 1
split 0,0,0 0,1,0 1
```

POLYOBJ

The world file command **polyobj** is used to create a static object in the world. This command creates the ceiling and floor of the court. The format of the polyobj command is:

```
polyobj points surface x1, y1, z1 ... x8, y8, z8
```

Points is the total number of data points that follow the command. Surface is the name of a color to paint the object. X1, y1, z1 through x8, y8, z8 are the data points for the polygon. There can be a maximum of eight vertices. The vertices are defined in long integer world coordinate format.

To create the ceiling of the court, we use the command:

```
polyobj 4 sgray  4000,2400,0  4000,240,4000  800,2400,4000  800,2400,0
```

Notice that there are no commas between the vertices. The floor of the court is created with the command:

```
polyobj 4 sgray  800,0,0  800,0,4000  4000,0,4000  4000,0,0
```

The polyobj is an easy way to create detail in the worlds. If you execute the demo program on the file court.wld, located on the enclosed disk, you see the red serving regulation lines on the floor of the court. These lines were created using the polyobj command.

Ball

The last object I will discuss is the ball. The file ball.plg is a 26 polygon ball for use in the court. When put into the world, the ball is expanded by a scaling factor of 5 and placed at the edge of the court.

Next

In the next **Working With Rend386** column, I will discuss the action of bouncing the wall up and down and against the racquetball court walls.

The Disk

As mentioned in the Preface, a diskette will be enclosed with each issue of PCVR. This disk will contain all of the programs in source form and most in executable form. This is being done to provide you with quick access to the information in the issues. This page will appear in each issue to describe the contents of the disk and which program relates to what article.

Building a Boom Head Tracker

adctest.exe	- This is a self-extracting file that creates the following files.
adc.pas	- This is the source code for the ADC0804 tester program.
adc.exe	- This program tests channel 0-2 of the ADC0804 circuit.
tracker.exe	- This is a self-extracting file that creates the following files.
headtrac.c	- This is source code for the head tracker demo program.
headtrac.exe	- This program demonstrates the boom head tracker.
hand.fig	- This is a figure file for use with headtrac.exe
yellcube.plg	- This is an object file for hand.fig.
redcube.plg	- ""
bluecube.plg	- ""
grencube.plg	- ""

Working With Rend386

workrend.exe	- This is a self-extracting file that creates the following files.
court.wld	- This is a racquetball court world for use with demo4.exe
courtwall.plg	- This is a wall object for use with court.wld.
ball.plg	- This is a ball object for use with court.wld.
line.plg	- This is a line object for use with court.wld.

Graphics

graphics.exe	- This is a self-extracting file that creates the following files.
axis4a.c	- This is source code for the individual rotation demo.
axis4a.exe	- This is the individual rotation demo.
axis4b.c	- This is source code for the transformation rotation demo.
axis4b.exe	- This is the transformation rotation demo.
egavga.bgi	- This is a .bgi file for EGA and VGA displays.

Last Page

Next Issue

The next issue of PCVR will be dedicated to 3D Sound. We will look at the technology currently available and determine which we can build ourselves.

Software

If you wish a diskette containing the software from all previous issues that did not include a diskette (1, 2, 3), send $5.00 and your diskette preference.

Submissions

We are always looking for suggestions, comments, and new ideas. To submit an article, send a diskette in any of the following formats.

1) ASCII with illustrations in GIF or PCX format
2) Word For Windows 1.1,1.2 or 2.0
3) Word Perfect 5.0,5.1

Address:

PCVR
1706 Sherman Hill Rd. #A
Laramie, Wyoming 82070

If we use your article, we will credit your account for the issue that the article appears.

I can be contacted at **gradecki@rodeo.uwyo.edu**. If you are interested in conversing about topics that appear in the magazine, drop me a line.